

Final Project

Dynamic Programming and  
Genetic Algorithm

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2018년 6월 24일

학부 정보통신공학과

학년 4학년

성명 김현진

학번 12121627



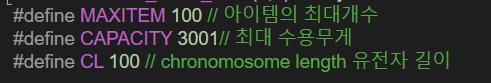
1. 개요

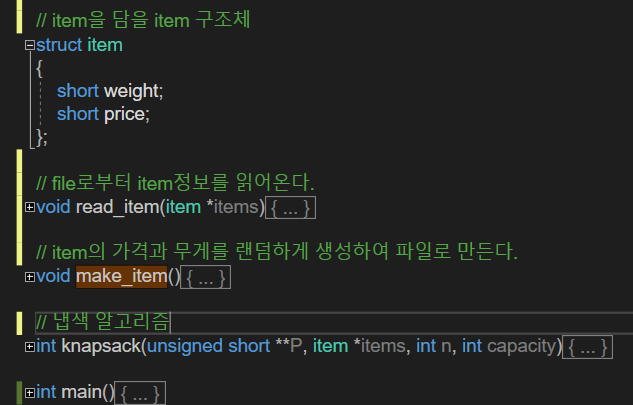
0-1 knapsack problem을 Dynamic Programming과 Genetic Algorithm으로 각각 풀어 본 뒤 성능 및 효과를 비교한다.

1. Dynamic Programming

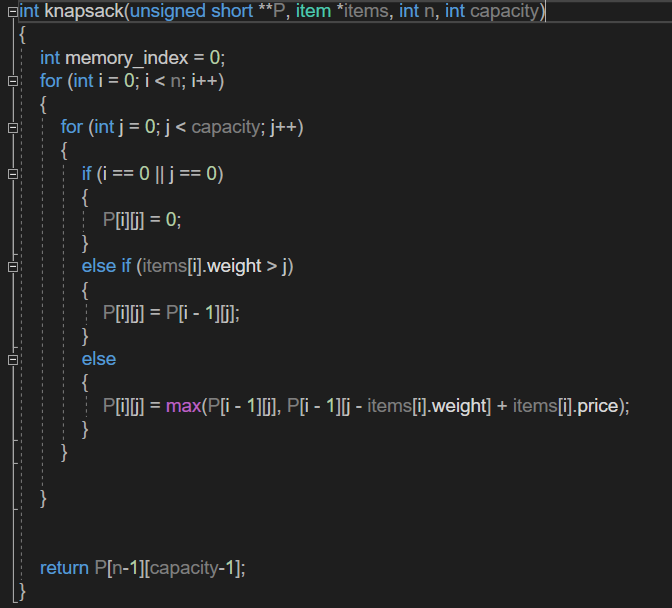
2.1 설계 상세

2.1.1 설계에 필요한 전체적인 구조

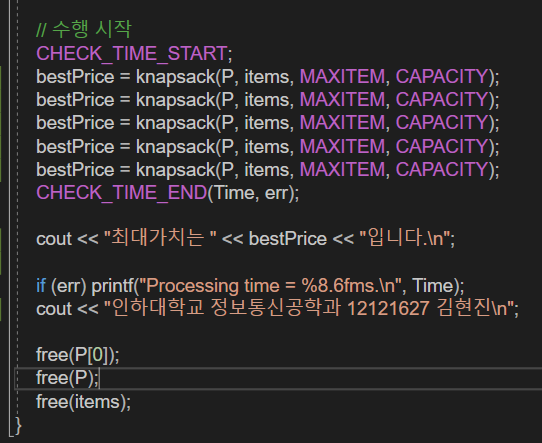


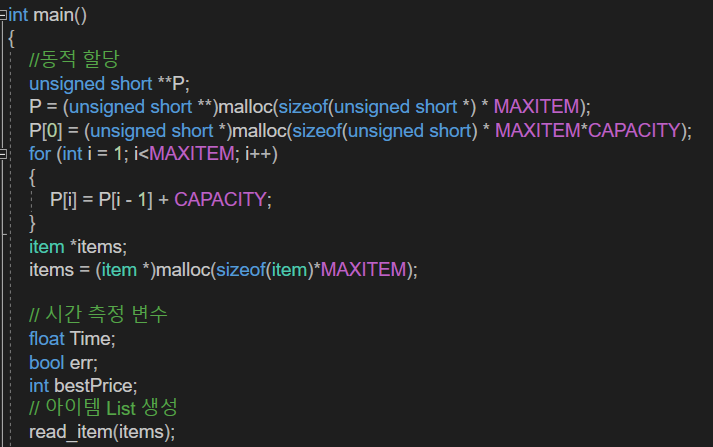


2.2.2 Knapsack algoritihm



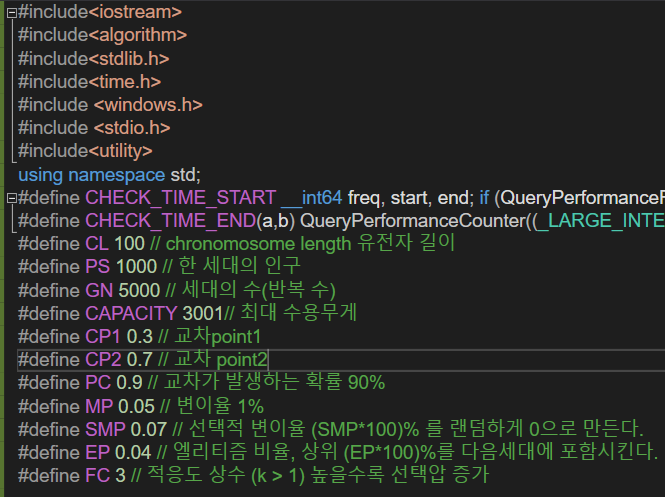
수업 때 배운 knapsack 알고리즘을 정석적으로 구현하였습니다.

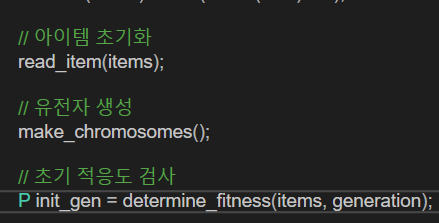
2.2.3 수행 개요

보다 수월한 시간 측정을 위해 5번의 knapsack 알고리즘 수행 시간을 기준으로 측정하였습니다.

3. Genetic Algorithm

3.1 설계 개요





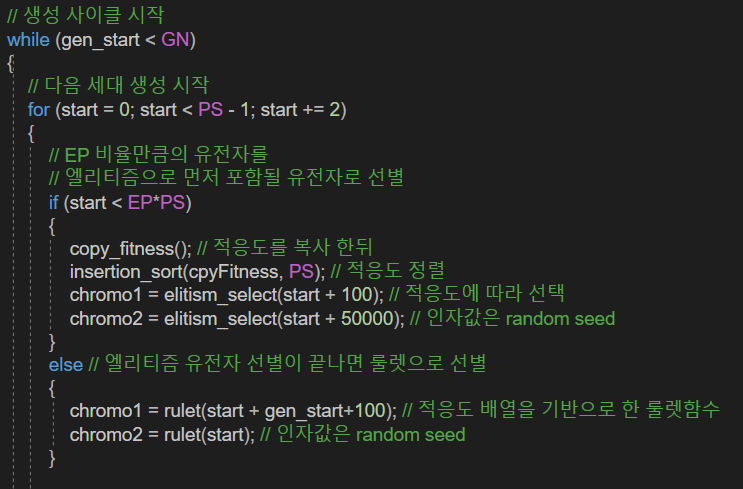
아이템을 읽어들인다. 실험을 진행할 초기 유전자를 생성한다. 랜덤하게 0과 1을 부여한다. 0이면 유전자 단백질(유전자를 형성하는 각각의 0과 1들) 인덱스 위치의 아이템이 포함되지 않은것이고, 1이면 포함된 것이다. determine\_fitness함수를 통해 적응도를 평가한다.



-> determine\_fitness 함수의 적응도 평가부분

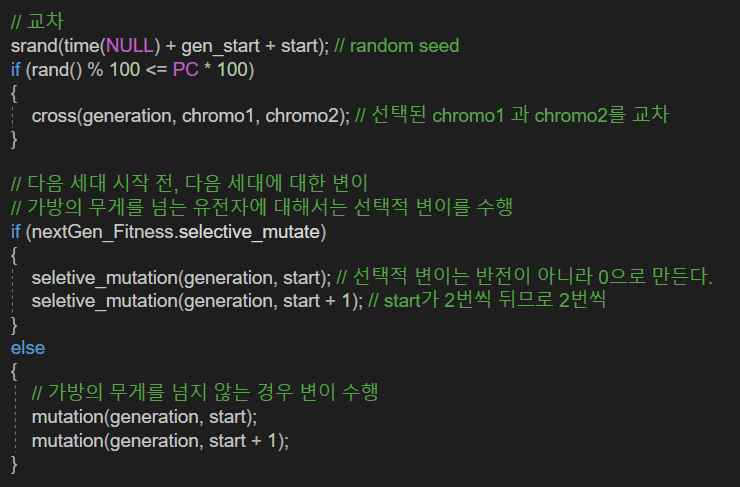
상단의 부분은 각 유전자들의 weight를 누적적으로 계산하여 배낭 무게를 넘는 유전자에 대해서는 선택적 변이 여부를 True로 하고 아닌 것들은 그냥 변이를 진행한다. 여기서 선택적 변이란 무게를 빠르게 최적화 하기위해 반전을 하는 변이가 아닌 무조건 0으로 바꾸는 변이이다.

하단의 부분이 적응도 검사이다. 적응도 함수는 FC(fitness constant)를 이용해 가장 큰 적응도를 가진 유전자가 가장 낮은 적응도를 가진 유전자의 FC배가 되게끔 설계하였고, 이때 에도 배낭 무게를 넘는 유전자의 적응도는 3/4로 줄임으로써 패널티를 주었다.

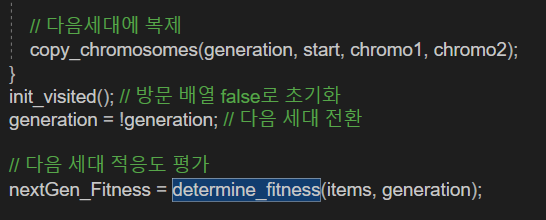


초기 적응도를 평가하고나면 생성 사이클을 시작한다.

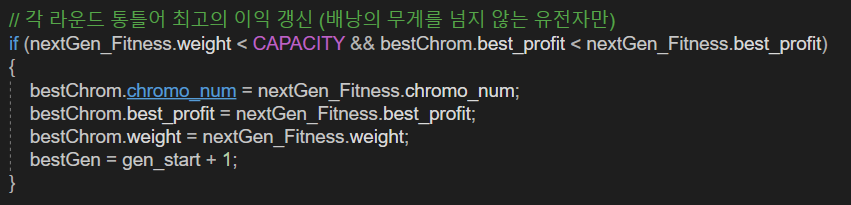
맨 처음에는 엘리티즘 함수를 이용해 먼저 선별될 유전자들을 선택한다. 그 후에는 룰렛 선택 방법을 통해 유전자를 선별한다.



유전자를 선별하고나면 교차를 수행한다. 교차확률은 90%로 정하였다. 선택된 chromo1, chromo2에 대해 교차를 진행한다. 교차가 끝 난 뒤 변이를 수행한다. 변이는 선택적 변이 여부에 따라 선택 변이와 그냥 변이로 진행한다.



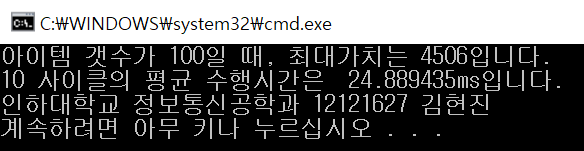
그 후에 다음세대로 복제를 한 뒤 다음 세대에 대한 적응도 평가를 수행한다.

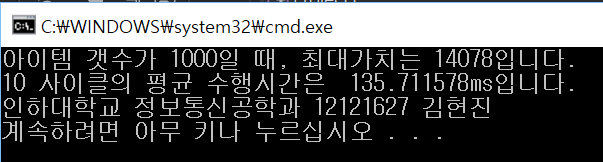


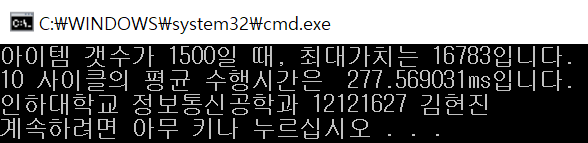
이후 각 세대별로 최고의 profit을 갱신하며 기록한다.

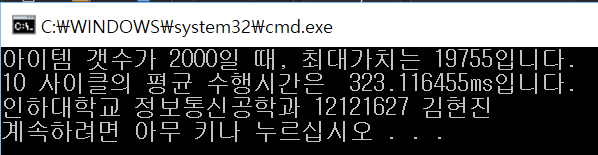
4. 성능 평가

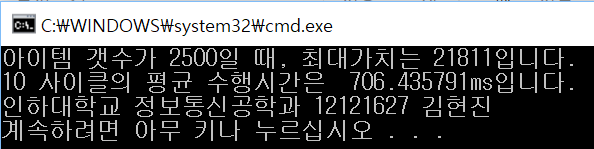
4.1 DP

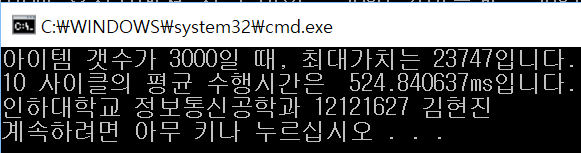


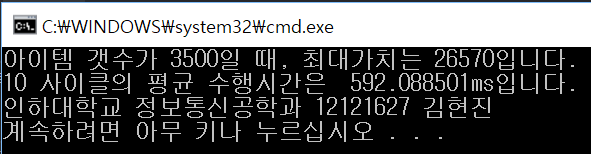


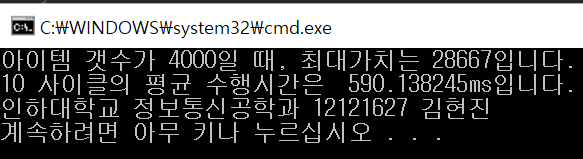


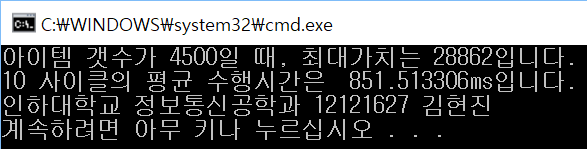


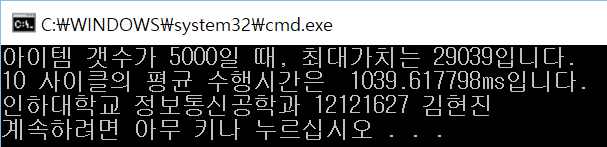




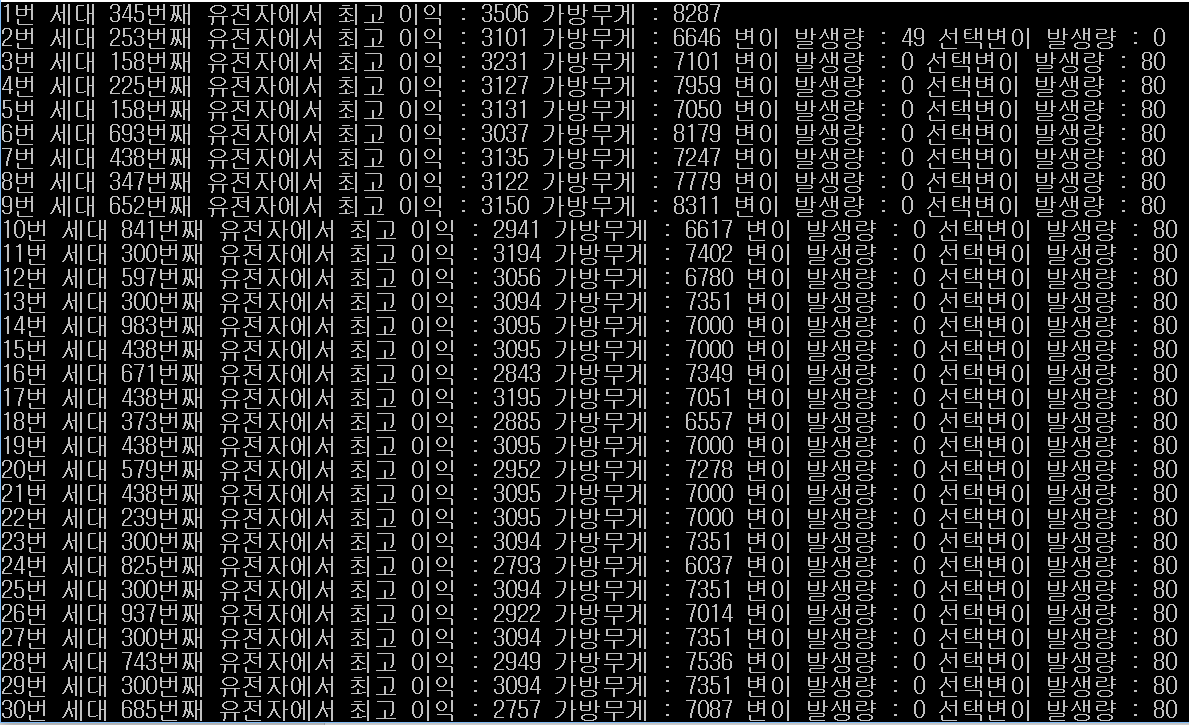








4.2 GA





150~200세대에서 무게가 현저하게 줄어든다.



무게에 대한 최적화는 300번대에서 거의 끝났다.

하지만 이익에 대한 최적화는 아직 낮다.



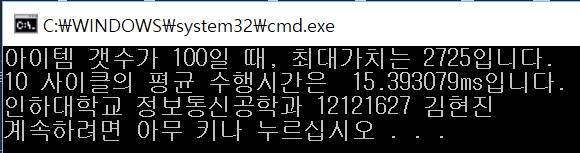
천번째 세대 돌입후에도 무게에 대한 최적도 변화는 거의 없다.



2천번째 진입시 최적도(이익)이 200가량 증가하였다.



계속해서 5000번까지 수행한 결과 2194라는 이익으로 최적화 되었다.



같은 item\_list.txt를 가지고 DP에서 수행결과 최적의 이익은 2725이다. 즉 GA에서 Global otimum을 찾지 못하고 Local optima를 찾는데 그쳤다. 수행시간에 대해서는, 5000번이나 반복하면서 찾기 때문에 한번 수행시 10분 이상 걸렸다. 이러한 작업을 각각의 변이율, 교차율, 엘리티즘 비율, 적응도 상수를 변화해 가면서 모두 기록해야 하는 것이 너무 버거워 일정상 진행 할 수 없었다.

5. 고찰

제네틱 알고리즘은 이론적으로 수업을 들을 때에도 매우 흥미로운 이론이였는데 직접 구현을 해보니까 더욱 신기했다. 푸는 방법을 몰라도 유전적 특성을 이용하여 해답을 찾는 것이 신기했다. 또한 각 세대를 반복하며 성능이 나아지는 알고리즘을 보며 내가 직접 창조한 창조물인듯한 인상을 받았고 어떻게 보면 내 자식이 성장하는 듯했다. 최적의 global optimum은 찾지 못했는데 이는 많은 변수, 상수, 변화율 등을 다채롭게 시도하지 못한 탓인 것 같다.

무게에 대해선 최적화가 되었지만, 가장 중요한 가격에대해선 최적화가 되지 않았다. 아마 무게를 최적화 함에 따라 유전자가 가격에 상관없이 많이 손상되었기 때문이라고 생각된다. 아마 global optimum을 찾는 최적의 환경조성이 있을것이다.